This is Jim’s source document for this Garage Door Opener Project.

The original code (derived from the article below) is included at the bottom of this document.

THE CURRENT CODE IS LOCATED AT [Link to Current Code](https://docs.google.com/document/d/11PyWrPhbZRAfl2MZz0g1ck-hqGb0CQvTGEFMSaRdAtM/edit?usp=sharing)

<http://arobito-docs.readthedocs.io/en/latest/mcu/experiments/pages/hc-sr04_distance_measurement.html>

# **Ultrasonic distance measurement with HC-SR04**

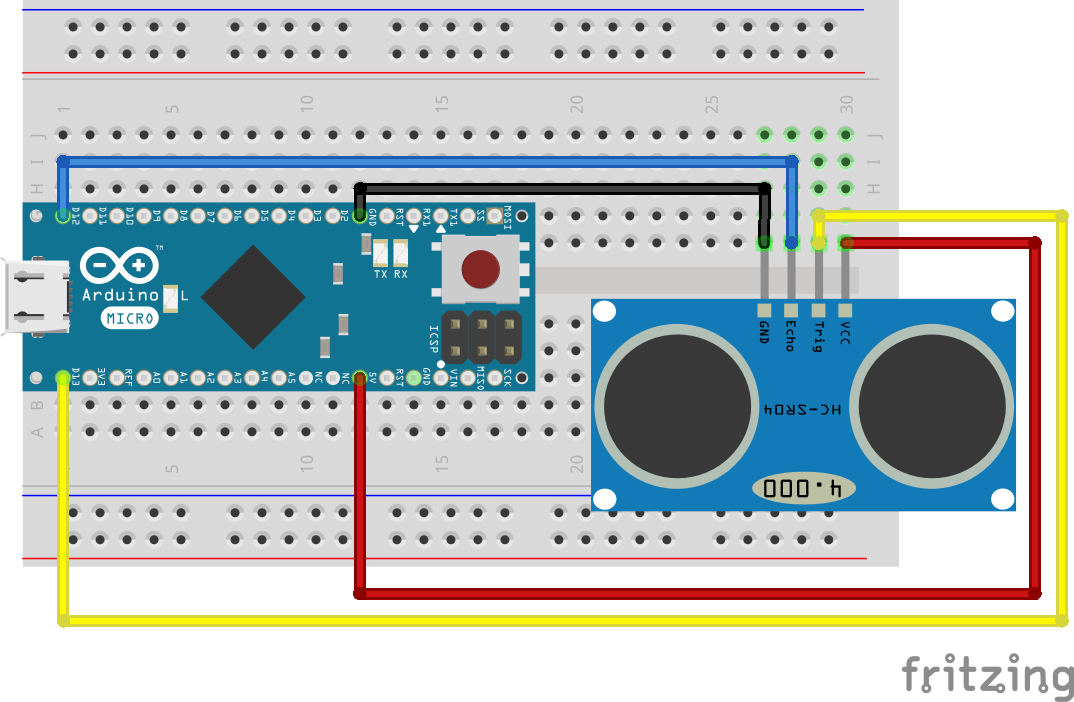
The HC-SR04 is a complete ultrasonic distance measurement device that is very simple to use and costs just about 3 EUR. The detection range as written in the data sheet is between 2 cm to 4 m, but experiments show that 3 cm to 2 m is more realistic.

## **Basic setup**

The basic setup consists of only these parts:

* a half-length breadboard
* an Arduino Micro v3
* an HC-SR04 ultrasonic distance sensor
* a few wires

The HC-SR04 works simply by sending a pulse to the trigger input and waiting for a pulse from the echo output. The length of this pulse allows the calculation of the distance to the detected object. We selected the Arduino pin 13 for sending the trigger pulse and pin 12 for receiving the echo pulse.



The power is delivered over the USB connection of the Arduino.

## **Programming the Arduino**

The code for this experiment can be found under [arobito/arobito-mcu/experiments/usonic/Basic\_Distance\_Measuring](https://github.com/arobito/arobito-mcu/tree/master/experiments/usonic/Basic_Distance_Measuring) as Atmel Studio Project.

The definitions:

*#define TRIGGER\_PIN 13  
#define ECHO\_PIN 12  
#define USONIC\_DIV 58.0  
#define MEASURE\_SAMPLE\_DELAY 5  
#define MEASURE\_SAMPLES 25  
#define MEASURE\_DELAY 250*

Trigger pin and echo pin are the two pins where the HC-SR04 is connected to. From the data sheet we know that we have to divide the measured microseconds by 58 to get centimeters. We delay our sample measurements by 5 µs, and take 25 samples to calculate the average. Between two measuring cycles the controller should wait for 1/4 second.

*void setup()  
{  
 // Serial monitoring  
 Serial.begin(9600);  
  
 // Initializing Trigger Output and Echo Input  
 pinMode(TRIGGER\_PIN, OUTPUT);  
 pinMode(ECHO\_PIN, INPUT);  
  
 // Reset the trigger pin and wait a half a second  
 digitalWrite(TRIGGER\_PIN, LOW);  
 delayMicroseconds(500);  
}*

During setup, the serial (USB) port get initialized and the trigger and the echo pin get assigned. To make sure we have a defined starting point, the trigger pin is put to low explicitly. After waiting half a second, the loop may start.

The loop simply triggers the measurement:

*void loop()  
{  
 delay(MEASURE\_DELAY);  
 long distance = measure();  
 Serial.print("Distance: ");  
 Serial.print(distance);  
 Serial.println(" mm");  
}*

The method measure() contains the sampling of single measurements:

*long measure()  
{  
 long measureSum = 0;  
 for (int i = 0; i < MEASURE\_SAMPLES; i++)  
 {  
 delay(MEASURE\_SAMPLE\_DELAY);  
 measureSum += singleMeasurement();  
 }  
 return measureSum / MEASURE\_SAMPLES;  
}*

One single measurement sequence looks like the following:

*long singleMeasurement()  
{  
 long duration = 0;  
 // Measure: Put up Trigger...  
 digitalWrite(TRIGGER\_PIN, HIGH);  
 // ... wait for 11 µs ...  
 delayMicroseconds(11);  
 // ... put the trigger down ...  
 digitalWrite(TRIGGER\_PIN, LOW);  
 // ... and wait for the echo ...  
 duration = pulseIn(ECHO\_PIN, HIGH);  
 return (long) (((float) duration / USONIC\_DIV) \* 10.0);  
}*

The programm’s output looks like the following:

Distance: 90 mm  
Distance: 90 mm  
Distance: 89 mm  
Distance: 92 mm  
Distance: 90 mm  
Distance: 197 mm  
Distance: 207 mm  
Distance: 1371 mm  
Distance: 209 mm  
Distance: 159 mm  
Distance: 125 mm  
Distance: 125 mm  
Distance: 127 mm  
Distance: 128 mm  
Distance: 124 mm  
Distance: 124 mm  
Distance: 125 mm  
Distance: 126 mm

Jim’s Code

#define TRIGGER\_PIN\_1 13

#define ECHO\_PIN\_1 12

#define TRIGGER\_PIN\_2 11

#define ECHO\_PIN\_2 10

#define BUZZER\_PIN 9

#define TONE\_PIN 8

#define USONIC\_DIV 58.0

#define MEASURE\_SAMPLE\_DELAY 5

#define MEASURE\_SAMPLES 25

#define MEASURE\_DELAY 250

#define ALARM\_TARGET\_1 10.0 // Target Distance

#define ALARM\_TARGET\_2 10.0 // Target Distance

#define ALARM\_DEADBAND 1.0 // Deadband

#define ALARM\_COUNT\_SETPOINT 4 // Number of loop to start alarming at

#define ALARM\_COUNT\_RESET 8 // Maximum Loops (counts) before reset

int ALARM\_COUNT\_1 = 0; // Current Loop Count

int ALARM\_COUNT\_2 = 0; // Current Loop Count

int CURRENT\_ALARM\_COUNT = 0;

float CURRENT\_ALARM\_TARGET = 10.0; // // Target Distance

// int mode = 1;

// int mode = 2;

int Current\_Sensor = 2;

int debug = 1;

int TRIGGER\_PIN = 0;

int ECHO\_PIN = 0;

float DISTANCE\_INCHES = 0.0;

void setup()

{

// Serial monitoring

Serial.begin(9600);

// Initializing Trigger and Buzzer Output and Echo Input

pinMode(TRIGGER\_PIN\_1, OUTPUT);

pinMode(ECHO\_PIN\_1, INPUT);

pinMode(BUZZER\_PIN, OUTPUT);

pinMode(TONE\_PIN,INPUT);

// Reset the trigger pin and wait a half a second

digitalWrite(TRIGGER\_PIN\_1, LOW);

delayMicroseconds(500);

// Initializing Trigger Output and Echo Input

pinMode(TRIGGER\_PIN\_2, OUTPUT);

pinMode(ECHO\_PIN\_2, INPUT);

// Reset the trigger pin and wait a half a second

digitalWrite(TRIGGER\_PIN\_2, LOW);

delayMicroseconds(500);

// Turn Alarm Buzzer output off

digitalWrite(BUZZER\_PIN, LOW);

delayMicroseconds(500);

}

void loop()

{

if (Current\_Sensor== 1)

{

Current\_Sensor = 2;

TRIGGER\_PIN = TRIGGER\_PIN\_2;

ECHO\_PIN = ECHO\_PIN\_2;

CURRENT\_ALARM\_COUNT = ALARM\_COUNT\_2 +=1;

CURRENT\_ALARM\_TARGET = ALARM\_TARGET\_2;

}

else if (Current\_Sensor== 2)

{

Current\_Sensor= 1;

TRIGGER\_PIN = TRIGGER\_PIN\_1;

ECHO\_PIN = ECHO\_PIN\_1;

CURRENT\_ALARM\_COUNT = ALARM\_COUNT\_1 += 1;

CURRENT\_ALARM\_TARGET = ALARM\_TARGET\_1;

Serial.println("---------------------------------------------------------------------------: ");

}

// CURRENT\_ALARM\_COUNT += 1;

delay(MEASURE\_DELAY);

// CURRENT\_ALARM\_COUNT += 1;

long distance = measure();

DISTANCE\_INCHES = distance/25.4;

// if (debug == 999)DISTANCE\_INCHES = 2222.2; //Force a value in each read

Serial.println(" ");

Serial.print("(");

Serial.print(CURRENT\_ALARM\_COUNT);

Serial.print(")");

Serial.print("- SENSOR ");

Serial.print(Current\_Sensor);

Serial.print(" --- DISTANCE: ");

Serial.print(DISTANCE\_INCHES);

Serial.print(" ---------------- ");

Serial.print(ECHO\_PIN);

Serial.print(" ");

Serial.println(TRIGGER\_PIN);

check\_alarm();

}

long measure()

{

long measureSum = 0;

for (int i = 0; i < MEASURE\_SAMPLES; i++)

{

delay(MEASURE\_SAMPLE\_DELAY);

measureSum += singleMeasurement();

}

return measureSum / MEASURE\_SAMPLES;

}

long singleMeasurement()

{

long duration = 0;

// Measure: Put up Trigger...

digitalWrite(TRIGGER\_PIN, HIGH);

// ... wait for 11 µs ...

delayMicroseconds(11);

// ... put the trigger down ...

digitalWrite(TRIGGER\_PIN, LOW);

// ... and wait for the echo ...

duration = pulseIn(ECHO\_PIN, HIGH);

return (long) (((float) duration / USONIC\_DIV) \* 10.0);

}

int check\_alarm()

{

// if (Current\_Sensor== 1)

// {

// CURRENT\_ALARM\_COUNT += 1;

// Serial.println("-----------------Count Incremented------------------: ");

// Serial.print(ALARM\_COUNT\_1);

// Serial.println(" ");

if (CURRENT\_ALARM\_COUNT > ALARM\_COUNT\_SETPOINT)

{

Serial.println(" ");

Serial.print("(");

Serial.print(CURRENT\_ALARM\_COUNT);

Serial.print(")");

Serial.print(" <1> Alarm Count ");

Serial.print(CURRENT\_ALARM\_COUNT);

Serial.print(" Greater than Setpoint ");

Serial.print(ALARM\_COUNT\_SETPOINT);

Serial.println(" ");

}

if (DISTANCE\_INCHES > (CURRENT\_ALARM\_TARGET + ALARM\_DEADBAND))

{

Serial.println(" ");

Serial.print("(");

Serial.print(CURRENT\_ALARM\_COUNT);

Serial.print(")");

Serial.print(" <2> DISTANCE\_INCHES ");

Serial.print(DISTANCE\_INCHES);

Serial.print(" GT ALARM\_TARGET ");

Serial.print(CURRENT\_ALARM\_TARGET);

Serial.print(" + DEADBAND ");

Serial.print(ALARM\_DEADBAND);

Serial.print(" (OK 2 ALM)------");

Serial.println(" ");

if (CURRENT\_ALARM\_COUNT > ALARM\_COUNT\_SETPOINT)

{

digitalWrite(BUZZER\_PIN, HIGH);

if (debug == 1)

{

Serial.println(" ");

Serial.print("(");

Serial.print(CURRENT\_ALARM\_COUNT);

Serial.print(")");

Serial.print(" \*\*\* <3> \*\*\*\*\*\* BUZZER ON ALARM \*\*\*\*\*\* SENSOR ");

Serial.print(Current\_Sensor);

Serial.print(" Distance: ");

Serial.println(DISTANCE\_INCHES);

}

}

}

if (CURRENT\_ALARM\_COUNT > ALARM\_COUNT\_RESET)

{

Serial.println(" ");

Serial.print("(");

Serial.print(CURRENT\_ALARM\_COUNT);

Serial.print(")");

Serial.print(" <4> ");

Serial.print(" ALARM COUNT ---- RESET ----- ---- RESET ----- ---- RESET -----: ");

Serial.print(CURRENT\_ALARM\_COUNT);

Serial.println(" ");

if (Current\_Sensor == 1)

{

ALARM\_COUNT\_1 = 0;

}

if (Current\_Sensor == 2)

{

ALARM\_COUNT\_2 = 0;

}

// CURRENT\_ALARM\_COUNT = 0;

digitalWrite(BUZZER\_PIN, LOW);

}

}